REMARKS

1. Claims 1, 3-4, 10, 12-13, 19, and 21-22 are Patentable Over the Cited Nemes

The Examiner rejected claims 1, 3-4, 10, 12-13, 19, and 21-22 as anticipated (35 U.S.C. §102(e)) by Nemes (U.S. Patent No. 5,893,120). Applicants traverse for the following reasons.

Independent claims 1, 10, and 19 concern processing an input file in a file system, wherein the input file has an input file name. A function is applied to map the input file name to a value. A data structure is processed to determine whether there is a preexisting file in the file system having a name that maps, according to the function, to the same value to which the input file name maps, wherein two files that map to a same value according to the function are capable of having a same name.

The Examiner cited col. 5, lines 55-68 of Nemes as disclosing the requirements of claims 1, 10, and 19. (Office Action, pg. 2) Nemes discusses applying a hash function to a unique value of a record to store and retrieve the associated record. (Nemes, col. 4, lines 53-67). The key of the record is translated into a hash table array that is used as an index into the array where searches for the data record begin (Nemes, col. 4, line67 to col. 5, line 5).

The cited part of Nemes discusses the removal of expired records in a linked list. The search key of the record being searched is hashed to provide a subscript that provides a pointer to the target linked list. Although the cited Nemes discusses a hashing function applied to a search key of a record, nowhere does the cited Nemes disclose the claim requirement of applying a function to a file name to produce a value. Instead, Nemes discusses applying the hash function to a search key of the record, but nowhere discusses applying the hash function to the name of the file as claimed.

Independent claims 1, 10, and 19 further require processing a data structure to determine whether there is a file in the file system having a name that maps, according to the function, to the same value to which the input filename maps, such that two files mapping to the same value according to the function have the same name. The cited Nemes discusses processing a list of hashed keys to search for a match and remove expired records from the list (Nemes, col. 6, line 5-34). However, nowhere does the cited Nemes disclose the claim requirement of processing a

data structure, such as a list, to determine whether there is a file in the file system having a name that maps to the same value to which the input file name maps. Instead, the cited Nemes is searching the list for a matching search key, not a matching file name as claimed. Nowhere does the cited Nemes disclose the claim requirement of processing a data structure to determine whether a file in the file system has a same name as an input file name.

Claims 1, 10, and 19 are patentable over the cited Nemes because Nemes does not disclose all the claim limitations.

Claims 3-4, 12-13, and 21-22 are patentable over the cited Nemes because they depend from claims 1, 10, and 19, which are patentable over Nemes for the reasons described above. Further, claims 4, 13, and 22 provide additional grounds of patentability over the cited art.

Claims 4, 13, and 22 depend from claims 3, 12, and 13, which require that the data structure includes an entry for each possible integer value capable of being generated from the hash function. Claims 4, 13, and 22 further require that processing the data structure to determine whether there is a preexisting file comprises determining whether the entry for the integer value to which the input file name maps indicates the presence of one preexisting file mapping to the same integer value as the input file name.

The cited col. 5, line 58 to col. 6, line 1 of Nemes discusses hashing the search key and comparing against elements in a list. If a match is found before reaching the end of the list, success is returned. Nowhere does this cited part of Nemes disclose the additional requirement of claims 4, 13, and 22 of determining whether the entry for the integer value to which the input file name maps indicates the presence of one preexisting file. Thus, claims 4, 13, and 22 require using the hash value to examine an entry in a data structure corresponding to the hash value to determine if a preexisting file name also maps to that hash value. The cited Nemes does not discuss this process of using the hash value as an index into the data structure to determine whether a file has a preexisting name as claimed, but instead uses the hash value, or subscript, to access a list and process through the list looking for a matching key and removing expired records. For these reasons, claims 4, 13, and 22 provide additional grounds of patentability over the cited art.

2. Claims 1, 6-8, 10, 15-17, 19, and 24-26 are Patentable Over the Cited Schmuck

The Examiner rejected claims 1, 6-8, 10, 15-17, 19, and 24-26 as anticipated (35 U.S.C. §102(e)) as anticipated by Schmuck (U.S. Patent No. 5,960,446). (Office Action, pg. 3) Applicants traverse for the following reasons.

The Examiner cited col. 7, lines 15-25 of Schmuck as teaching the requirements of claims 1, 10, and 19. Schmuck discusses the use of hashing to provide for very fast insert, delete and lookup, as well as a sequential scan of a large set of data records. (Schmuck, col. 7, lines 7-13) The cited col. 7, lines 12-17 discusses the use of a cursor value to guarantee that a sequential scan will not return duplicate records and not records that were inserted or deleted while the scan was in progress. The further discussion in col. 7 discusses how to use hash buckets if the number of records to be hashed is not known in advance and may expand.

Although the cited col. 7 of Schmuck discusses a hash list to index a large set of data records, nowhere does Schmuck disclose the requirement of independent claims 1, 10, and 19 of processing a data structure to determine whether there is a preexisting file in the file system having a name that the function maps to a same value, such that two files mapping to a same value may have a same name. Instead, the cited section of Schmuck primarily concerns how to use buckets to extend the number of available hash numbers.

Accordingly, claims 1, 10, and 19 are patentable over the cited Schmuck because Schmuck does not disclose every claim limitation.

Claims 6-8, 15-17, and 24-26 are patentable over the cited Schmuck because they depend from claims 1, 10, and 19, which are patentable for the reason discussed above and for the following additional reasons.

Claims 6, 15, and 24 depend from claims 1, 10, and 19, respectively, and further require applying the function to each file name in the file system to map each file name to one value and indicating in the data structure, for each file name, that there is one preexisting file for the value to which the file name maps.

The Examiner cited col. 7, lines 15-25 of Schmuck as disclosing the additional requirements of claims 6, 15, and 24. The cited section of Schmuck discusses using hashing to

index a large set of records and how to extend the hash table using buckets. However, nowhere does the cited Schmuck disclose the claim requirement of indicating in the data structure that there is one preexisting file for a value to which the file names map. Thus, Schmuck does not disclose the additional requirements of claims 6, 15, and 24. Accordingly, claims 6, 15, and 24 provide further grounds of patentability over the cited art.

Claims 7, 16, and 25 depend from claims 6, 15, and 24 and further require that the input file is the subject of an access request. Further, each file in the file system is scanned to determine if there is at least one preexisting file having the same name as the input file name if there is one preexisting file in the file system having a name that maps, according to the function, to the same value to which the input file name maps. The Examiner cited col. 9, lines 35-53 against the additional requirements of claim 7, 16, and 25.

The cited col. 9, lines 35-53 discusses computing a hash value from a key and searching hash buckets to find a hash value. If a matching hash value is found, then the record for that hash value is found and returned, else not found is returned. Claims 7, 16, and 25 require using the function to map an input file name to a value and determine if one preexisting file has the same name as the input file. Nowhere does the cited discussion of hashing described in Schmuck disclose the claim requirement of determining whether one preexisting file has the same name as the input file subject to the access request by determining whether one preexisting file has a name that maps to the same value as the input file name. Instead, the cited col. 9 of Schmuck discusses a basic lookup operation in a hash table to look for a record whose search key has a matching hash value not determining whether a preexisting file has a same name as an input file subject to an access request as claimed.

Moreover, claims 7, 16, and 25 additionally require scanning each file in the file system to determine if there is at least one preexisting file having the same name as the input file if the name of one preexisting file maps, according to the function, to the same value to which the input file name maps. Nowhere does the cited Schmuck suggest performing the additional step of scanning the file system for a matching file name if the hash table has a match to the hashed

key. Instead, the cited col. 9 of Schmuck just returns the record if there is a match, nowhere does the cited Schmuck go out and scan a file system as claimed upon detecting a match.

Accordingly, claims 7, 16, and 25 provide additional grounds of patentability over the cited art.

Claims 8, 17, and 26 depend from claims 7, 16, and 25, which require that the input file is the subject of an access request. Claims 8, 17, and 26 additionally require that the request to access the input file is to add the input file as a new file. The input file is added as a new file to the file system if no preexisting file in the file system has the same name as the input file name the access request is rejected if there is a preexisting file in the file system having the same name. The Examiner cited col. 9, lines 35-64 of Schmuck against the additional limitations of claims 7-8, 16-17, and 25-26.

The cited col. 9 of Schmuck discusses a lookup operation and insert operation for a record having a given input key. The cited Schmuck mentions that if a record with the given key of the record to insert already exists in the hash bucket, an "already exists" error is returned. Although Schmuck discusses making sure that no record having the same hash value has been inserted as the hash value of the record to insert, the claims require using the hash operations in the context of determining whether to add a file to a file system, such that the input file is added to the file system if there is no preexisting file having the same name. Nowhere does the cited Schmuck disclose using the hash table to determine whether to insert a file having a file name in the file system by considering whether a preexisting file having the same name as the file to insert exists in the system as claimed.

Accordingly, claims 8, 17, and 26 provide additional grounds of patentability over the cited art.

3. Claims 2, 5, 9, 11, 14, 18, 20, 23, and 27 are Patentable Over the Cited Art

The Examiner rejected claims 2, 5, 11, 14, 20, and 23 as obvious (35 U.S.C. §103) in view of Nemes and cited additional prior art for the limitations added in the dependent claims. (Office Action, pgs. 4-5) Applicants submit that claims 2, 5, 11, 14, 20, and 23 are nonetheless

patentable over the cited art because they depend from claims 1, 10, and 19, which are patentable over the cited art for the reasons discussed above.

Claims 9, 18, and 27 depend from claims 7, 16, and 25 and further require that the access request to the input file is to update a file in the file system with data from the input file. A preexisting file is updated in the file system having the same name as the input file with the data in the input file if there is such a preexisting file. The access request is rejected if there is no preexisting file in the file system having the same name as the input file name.

The Examiner rejected claims 9, 18, and 27 as obvious (35 U.S.C. §103) in view of Schmuck. (Office Action, pg. 6) Applicants submit that claims 9, 18, and 27 are patentable over the cited art because they depend from base claims 1, 10, and 19 as well as intervening claims, which are patentable over the cited art for the reasons discussed above. Applicants further submit that claims 9, 18, and 27 provide additional grounds of patentability over the cited art.

In rejecting claims 9, 18, and 27, the Examiner recognized that the cited Schmuck did not disclose the additional requirements of updating a preexisting file if there is a preexisting file having the same name as the input file. The Examiner found that such a modification of Schmuck would be obvious because Schmuck produces an "already exists" message if an insert operation is attempted on a record that already exists, i.e., the hash table indicates that a record having the same hash value as the record to insert already exists.

Applicants traverse this finding that it would be obvious to apply an update because Schmuck nowhere teaches or suggests the claim requirement of updating a record if the hash table indicates that a record already exists. Instead, the cited Schmuck only concerns inserting a record and not updating a preexisting file as claimed. Accordingly, claims 9, 18, and 27 provide additional grounds of patentability over the cited art.

4. The Added Claims 28-36 Are Patentable Over the Cited Art

Added claims 28, 31, and 34 depend from claims 1, 10, and 19 and further require searching the file system for one file having the same name as the input file name if the data

structure indicates that one preexisting file has a name that maps, according to the function, to the same value to which the input file maps. An operation is performed if the file system includes one file having the same name as the input file.

The additional requirements added in claims 28, 31, and 34 are disclosed at page 8, lines 3-12 and FIG. 2, blocks 110-114 of the Application.

Applicants submit that claims 28, 31, and 34 are patentable over the cited art because the Examiner has not cited any art that teaches or suggests searching the file system if the data structure indicates that one preexisting file has a name that maps, according to the function, to the same value to which the input file maps.

Added claims 29, 32, and 35 depend from claims 28, 31, and 34, and further require that performing the operation comprises applying update data to the file having the same name as the input file if the file system includes one file having the same name as the input file.

The additional requirements added in claims 29, 32, and 35 are disclosed on page 8, lines 12-15 of the Application.

Added claims 30, 33, and 36 depend from 28, 31, and 34, wherein the input file comprises a file to add to the file system. An error is returned if the file system includes one file having the same name as the input file. The input file is added to the file system if the file system does not include one file having the same name as the input file.

The additional requirements added in claims 30, 33, and 36 are disclosed on page 8, lines 5-10 and FIG. 2 of the Application.

Applicants submit that claims 29, 30, 32, 33, 35, and 36 are patentable over the cited art because they depend from one of claims 28, 31, and 34, which are patentable over the cited art for the reasons described above, and because the Examiner has not cited any art teaching the additional requirements of claims 29, 30, 32, 33, 35, and 36.